A Tool for Management of Knowledge Dispersed throughout Multiple References

Carlos Sáenz-Adán[1], Francisco J. García-Izquierdo[1], Ángel Luis Rubio[1],
Eduardo Sáenz-de-Cabezón Irigaray[1], Emilio Rodríguez-Priego[1] and Oscar Díaz[2]
[1]*Departamento de Matemáticas y Computación, Universidad de La Rioja, Logroño, Spain*
[2]*Departamento de Lenguajes y Sistemas Informáticos, Universidad del País Vasco, San Sebastián, Spain*

Keywords: Model Analysis, Model Tools, References-enriched Concept Map, Natural Language Processing.

Abstract: When modeling tasks are performed, it is important that the different modeling team members share a common vocabulary. This implies not only agreement on the terminology itself but especially on the meaning of the terms used. To this end it comes in handy to have graphical tools for sharing and analyzing the knowledge dispersed throughout different sources. In this paper we present RCMTool, a tool for creating References-enriched Concept Maps (RCM). This technique has been specially designed to facilitate the compact presentation and comparison of different definitions provided by multiple authors in diverse sources. This paper presents the main features of RCMTool, based on the development of an RCM metamodel and the inclusion of a natural language processing engine.

1 INTRODUCTION

A clear delimitation of the particular nounances in the work context is essential for the correct development of any modeling task, especially when this task is carried out jointly by teams of medium or large size. However, the available information can be so broad that the specification of the context often becomes a complex task in itself. In particular, the simple act of determining a vocabulary whereby the definition of terms is accepted by the whole team is an important task. The use of dictionaries, thesauri and encyclopaedias (Wikipedia being a case of note) is helpful but it does not solve the problem. For instance, in previous experiences related to the development of a tool for business project management, the authors had to make a great effort to reach an agreement about what was the meaning and features of the term "business process" for the team. The problem is even more pronounced in the field of research, in which different papers may have different and even contradictory definitions of the same concept. Typically, not all of them are equally relevant to determine the meaning of the discussed term. The author prestige or the influence of one of the definitions in the literature makes some defintions stand out from other, which may make the nouenances of the meaning of the term are closer to one or the other.

One way to alleviate this problem is to provide tools that allow a visual and compact representation of multiple definitions. In this paper we present RCMTool, a tool that aims to perform this task either individually or collaboratively by means of the implementation of the References-enriched Concept Map (RCM) technique (Rodriguez-Priego et al., 2013).

In (Rodriguez-Priego et al., 2013) RCM is defined as a Concept Map that "allows the visualization of a set of definitions about a term discussed in the literature, facilitating the analysis of such definitions in relation to the authors who propose them. The appearance of an RCM resembles a Concept Map since it is basically a diagram showing relationships (links) between concepts. Nevertheless, RCMs enrich the links with references to the publications".

An RCM allows the graphical comparison of different propositions through the *Path* feature. Fig. 1 depicts a portion of the RCM from Fig. 5 to illustrate its components. In this figure we can see four different definitions of "Language" from different sources. For instance, one of this definitions corresponds to the *Path* "Language is a system of signs expressing ideas" (highlighted in the figure). This Path is a sequence of *Concepts* such as "language"(main concept), "system", "signs" and "ideas", graphically represented with blue boxes. These *Concepts* are connected by *Linking Phrases* -words between blue

Saénz-Adán C., García-Izquierdo F., Rubio Á., Irigaray E., Rodríguez-Priego E. and Díaz O..
A Tool for Management of Knowledge Dispersed throughout Multiple References.
DOI: 10.5220/0005499700790086
In *Proceedings of the 10th International Conference on Software Paradigm Trends* (ICSOFT-PT-2015), pages 79-86
ISBN: 978-989-758-115-1
Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

boxes ("is", "of" and "expressing")-. A *Path* must always be associated with a corresponding bibliographical *Reference*, so that the source of the information can be traced down (in the example this is the reference labeled with the number 3). In addition, an RCM allows to have *Concepts* to be organized in *Layers* (boxes with different shades of blue), so that the creator of the diagram assigns degrees of relevance to the *Concepts*, depending for example on the proximity of each concept in the map to the main concept whose meaning he/she is attempting to clarify. All of these features ensure that an RCM is a type of model that can represent knowledge gathered from definitions that are dispersed throughout multiple sources.

The RCMTool pursues three objectives. First, it allows the management of bibliographical references. These sources can be inserted manually, also through files (in BIBTEX format) or through integration with reference management tools like Mendeley. Second, RCMTool facilitates the graphical representation of definitions through paths. A Path can be manually constructed (graphically with building blocks of RCM) or created in a semi-automated way by a restricted natural language processing assistant. Finally, the tool should allow collaborative use, enabling several creators of an RCM to work on the same diagram providing new definitions or references or improving those already existing.

RCMTool is a web tool developed using consolidated technologies such as HTML5/CSS3, NodeJS, JS, oAuth, and REST APIs. It uses Rappid (Client IO, 2014), a toolkit for building interactive diagramming applications, under an Academic User License.

The paper is structured as follows. Section 2 shows the related work. Sections 3 and 4 explain the fundamentals and features of the tool. RCMTool at work is shown in Section 5. The last section sets out our conclusions and further work.

2 RELATED WORK

Various techniques for knowledge representation exist in the literature. Many of these techniques are based on the graphical representation of ideas, concepts, topics or terms (in many cases using graphs). Topic Maps (Park and Hunting, 2003) and Mind Maps (Buzan, 2006) are two cases in point. Mind Maps has tools, such as FreeMind (FreeMind Dev. Team, 2014) or XMind (XMind Ltd., 2014) that allow the computer-based creation of that kind of maps. On the other hand, Concept Maps (Novak and Cañas, 2006) stands out for its extensive use. Concept Maps were created by Novak, based on Ausubel's theory of meaningful learning (Ausubel, 1963; Ausubel et al., 1968).

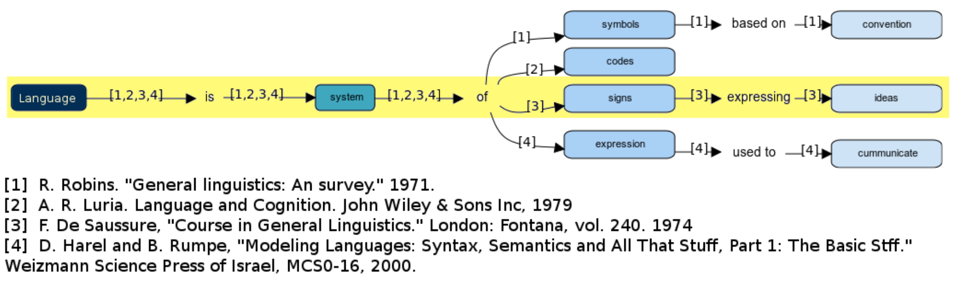
When developing modeling tasks, however, often what is important is not only the representation of knowledge but a strong consensus on the concepts and terms used. This consensus facilitates the modeling task since ambiguities and misunderstandings among members of the development team are avoided. We found that although it might be feasible to use Concept Maps (or other techniques) in order to compare and to present definitions in a compact way, none of the existing techniques are perfectly suited to this task. On the one hand, the goal of a Concept Map is not to present *definitions of concepts*, but to represent *knowledge about concepts*. On the other hand, a Concept Map does not facilitate comparison of different definitions available from various sources, nor does it have a precise system for managing these sources.

Inspired by Concept Maps, the References-enriched Concept Maps (RCM) technique (Rodriguez-Priego et al., 2013) has been developed to include several features, mainly *path labeling* and *concept layering*, which meet the needs explained above. The RCM technique also includes additional features to measure the complexity of the included definitions and their relationship with their corresponding references. These features allow authorship attribution to be mantained. This information, in turn, allows the RCM creator to consider the relative weight of different definitions (e.g. a definition included in an online encyclopaedia could be considered less relevant than another included in an article in a prestigious scientific journal), representing a certain kind of knowledge modeling.

CMapTool is also of interest (Cañas et al., 2005). It focuses on Concept Map, emphasizing its information visualization features. This tool is not tailored to the specific features of RCM. It allows the association of information resources to concepts or linking words on the map, thanks to which a Concept Map can be used as a repository of information about a certain topic. However, it does not allow the association of resources to sequences of concepts and linking phrases (paths). Since the above feature is not available in any system, yet of paramount importance, so it has been necessary to develop a tool that realizes it.

3 TOOL FUNDAMENTALS

The tool has been designed keeping in mind two fundamental principles. Firstly it clearly distinguishes the graphical representation from the conceptual model behind an RCM. In a similar way



[1] R. Robins. "General linguistics: An survey." 1971.
[2] A. R. Luria. Language and Cognition. John Wiley & Sons Inc, 1979
[3] F. De Saussure, "Course in General Linguistics." London: Fontana, vol. 240. 1974
[4] D. Harel and B. Rumpe, "Modeling Languages: Syntax, Semantics and All That Stuff, Part 1: The Basic Stff." Weizmann Science Press of Israel, MCS0-16, 2000.

Figure 1: Portion of RCM shown in Figure 5.

to (Kowata et al., 2010), who propose a metamodel for Concept Maps, we have developed an RCM metamodel. Using this approach, each particular RCM is a model, since it is an instance of a metamodel. This instance is linked internally with the graphical representation drawn by the tool. Secondly, the tool has a natural language processing engine. This engine can largely automate the construction of RCMs, since when a restricted natural language text definition is typed, the tool is able to draw the path corresponding to that definition.

3.1 Metamodel

Fig. 2 shows the core of the RCM metamodel. This metamodel is completed with a series of Object Constraint Language (OCL).

The two basic building blocks (*RCMNodes*) of any RCM are *Concept* and *LinkingPhrase*. With these basic elements, *Paths* are constructed as an ordered list of *Concepts* connected through *LinkingPhrases*. In order to precisely define the notion of Path required by RCM it is necessary to include the technical notion of *Syntagm*. Thus, a *Path* is actually a list of *Syntagms* (which in turn are formed joining a *Concept* and a *LinkingPhrase*). The semantic construction of *Paths* is completed with the following OCL constraints:

```
context Path
inv: syntagmList->
        excluding(syntagmList->first())
        ->forAll(elements.size()=2)
inv: (syntagmList->first()).elements->
        size()=1
inv: (syntagmList->first()).elements.
        oclIsKindOf(Concept)
```

These rules set up the facts that “Every Syntagm in syntagmList from a Path has two elements except the first” and that “The first Syntagm in syntagmList has only one element, and this element is a Concept”.

Other relevant properties are represented in the RCM metamodel. Both *Concepts* and *LinkingPhrases* may be included in several different *Paths*. This property enables the comparison of definitions collected from different sources, by calculating metrics of similarity between groups of definitions. Furthermore, *Concepts* are assigned to *Layers*, following the criteria chosen by the RCM creator. Finally, every *Path* has a *Reference* associated with it, and one *Reference* may include several definitions (and therefore have several associated *Paths*).

3.2 Natural Language Processing

According to (Novak, 1984) it is essential to isolate concepts from linking words (LinkingPhrases in RCM). Both are important as language units, but they play different roles in the transmission of meaning. While concepts are labeled by words that represent things and events, relationships are a way of linking two concepts in a propositional form.

With the aim of processing the mark up of a word in a text as corresponding to a particular part of speech, based on both its lexicon, as well as its context, we use *part-of-speech tagging* (POS tagging or POST). This technique (also called grammatical tagging or word-category disambiguation) has been used for tagging words in accordance with the part of the speech tag-set used by the Penn Treebank Project (Santorini, 1990) (see Table 1).

We could consider Noun Phrase (NP), Verb Phrase (VP) and Prepositional Phrase (PP) as primary candidates to be mapped as RCMNodes. Noun Phrase (NP) is the most important candidate to be a concept; Verb Phrase (VP) is aspirant to be a LinkingPhrase, and Prepositional Phrase (PP) could be both a concept, taking the essence of the Noun Phrase, and a relationship, if we take into consideration the preposition that appears before the Noun Phrase.

The implemented algorithm is based on (Brill, 1995), and it is used to apply transformation rules (some examples of these rules are shown in Table 2). This algorithm tags words using the rules and following the schema in Fig. 3. The original, unannotated text moves to a second state called “initial state”. In this second state words are tagged in four steps which are independent of the context: first, tag words from

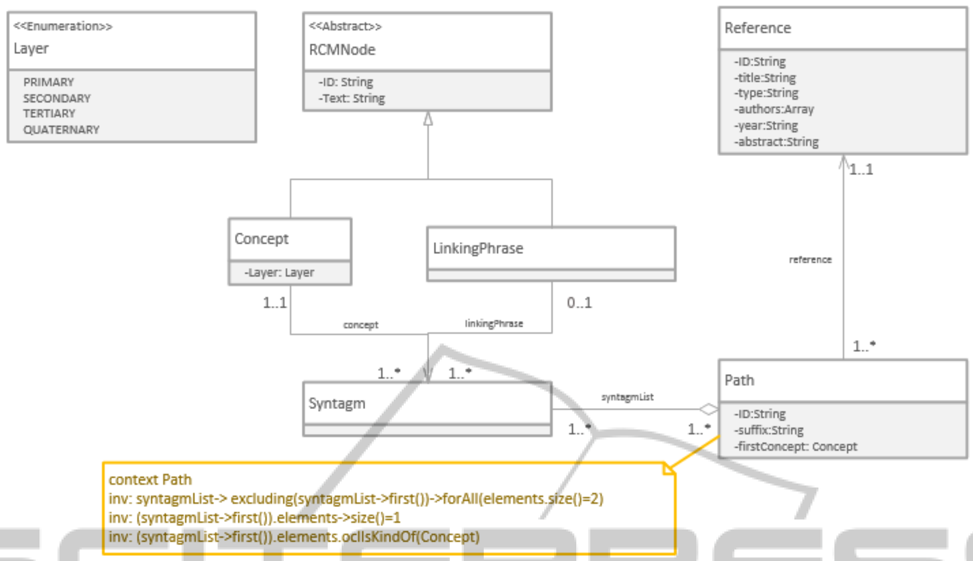


Figure 2: RCM Metamodel.

Table 1: List of tags of corresponding part of speech used by Penn Treebank project.

Number	tag	Description	Number	tag	Description
1	CC	Coordinating conjunction	2	CD	Cardinal number
3	DT	Determiner	4	EX	Existencial there
5	FW	Foreign word	6	IN	Preposition or subordinating conjunction
7	JJ	Adjective	8	JJR	Adjective, comparative
9	JJS	Adjective, superlative	10	LS	List item marker
11	MD	Modal	12	NN	Noun, singular or mass
13	NNS	Noun, plural	14	NNP	Proper noun, singular
15	NNP	Proper noun, plural	16	PDT	Predeterminer
17	POS	Possesive ending	18	PRP	Personal pronoun
19	PRP$	Possesive pronoun	20	RB	Adverb
21	RBR	Adverb, comparative	22	RBS	Adverb, superlative
23	RP	Particle	24	SYM	Symbol
25	TO	to	26	UH	Interjection
27	VB	Verb, base form	28	VBD	Verb, past tense
29	VBG	Verb, gerund or present participle	30	VBN	Verb, past participle
31	VBP	Verb, non-3rd person singular present	32	VBZ	Verb, 3rd person singular present
33	WDT	Wh-determiner	34	WP	Wh-pronoun
35	WP$	Possessive Wh-pronoun	36	WRB	Wh-adverb

a dictionary; second, tag words from a suffix; third, tag words as NN if they are not the first word and they are capitalised; and finally transform numbers and tag as CD. After this step the text is annotated, although some errors can occur. These errors are corrected with transformation rules (Table 2) which are context-dependent.

Example. Consider the statement “Language is a set of sentences”. This proposition has been fragmented and tagged in the initial state as follows :

- Language/ NN
- is /VB
- a /DT
- set /VB
- of /IN
- sentences /NNS

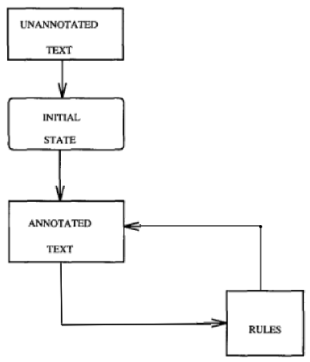


Figure 3: Flow for tagging based on (Brill, 1995).

Table 2: Some transformation rules.

From	To	Condition
-	NN	Before a MD
NN	VB	After a RB
JJ	RB	Before a JJ
VB	NN	After a DT
VB	JJ	After CP and RB
VB	JJ	After DT before NN

The tagging task is performed without context and has errors, since the word "set" has been tagged as VB despite the fact that in this example "set" means a number or combination of things of similar nature or function, and is therefore a noun. This error is corrected by means of the application of transformation rules in the last step. There is a rule that states that if a word is a VB and the preceding word is a DT, the first word will be tagged as NN. Hence, as "set" is preceded by a DT it has been tagged to NN.

4 TOOL DESIGN

The design of the tool incorporates aspects related both to functionality and the user interface.

4.1 Features

RCMTool was developed with several key features (Fig. 4) in mind. Naturally, and since it is a diagramming tool, addition and deletion of the building blocks of RCM (namely Concepts and Linking phrases) and editing of their properties are allowed. Linking of elements is no longer immaterial, since only concepts with linking phrases (in either direction) must be connectable. It makes no sense that two concepts (or linking phrases) are connected to each other, since by definition of RCM must be possible traverse a path in a natural way and read a meaningful statement. Furthermore, the user must be able to edit link structure's and labels, which are lists of reference identifiers. One of the key features of RCM is the creation of paths. Several elements (both concepts and linking phrases) can be selected simultaneously and can be then marked as a path. This path corresponds to a statement that appears in a bibliographical reference, so that both path and reference are related. The computation of the metrics for determining the complexity of an RCM and the collaborative use of the tool are other features that should be included. Finally, it is important that the user can export RCMs to other formats, both as graphics (such as JPEG) and other formats based on data processing (such as JSON). This export feature facilitates connectivity and interoperability with other tools. Thus RCM can be incorporated into other documents or analyzed by specific tools.

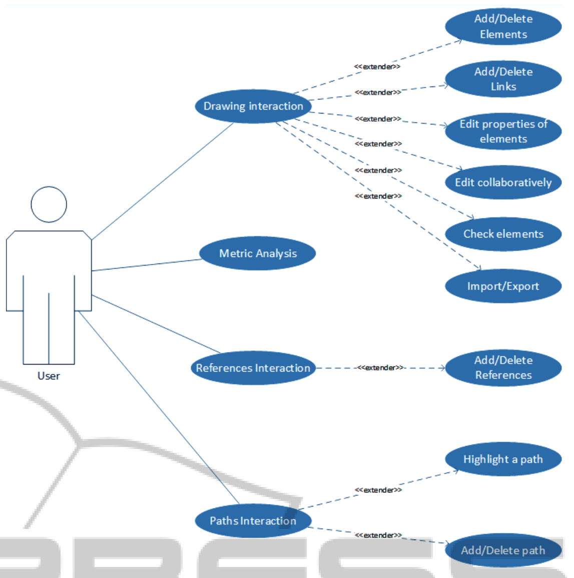


Figure 4: Functionality of the tool.

4.2 User interface

The interface is split into areas in line with the functionality of the tool (see Fig. 5).

1. Toolbar. Located at the top of the interface. It allows users to add anonymous propositions, add a list of propositions with their corresponding references, add new references to the references list, highlight a path, import/export the RCM, clear the canvas and create an image of the RCM.
2. Stencil. Located on the top left of the interface and groups the main building blocks of an RCM (Concepts and LinkingPhrases). The user can add them by dragging them onto the workspace.

3. References list. Located at the left of the interface and shows the available references. Each reference has an identifier, a list of authors and the title of the article or the URL.
4. Paths list. Located on the bottom left of the interface. It shows the existing paths, each path has an identifier linked with the corresponding reference.
5. Workspace. it is the main editing area, located in the centre of the interface.
6. RCMs list. Located at the top right of the interface and shows all the RCMs shared by the user.
7. Inspector. Located on the right of the interface. It is used to display and edit properties of elements.
8. Status bar. Located at the bottom of the interface. It displays information about the application state.

5 TOOL USAGE

The tool has been developed as a web application, based on the metamodel explained in Section 3.1, implementing the features listed in Section 4.1 and following the user interface design described in Section 4.2. The tool also includes a natural language processing engine that uses the algorithm described in Section 3.2.

5.1 Drawing Interaction

There are three different ways to add elements in order to build an RCM:

- The user can create a branch of the graph through an anonymous (i.e. without an associated reference) textual statement.
- Branches of the graph can be created through a text containing statements associated with their references.
- The user can add new elements by manually dragging them from the stencil to the workspace and then editing the text property.

In the two first cases, the tool uses the natural language processing engine (described in Section 3.2). Although this process splits sentences into concepts and linking phrases, it does not generate a final result in all cases, but provides powerful support for the construction of the graph, which can be manually edited by the RCM creator. It is important to note that, since each concept is unique and all the elements in the graph must be associated with at least one reference (see Fig. 2), the elements that temporarily have no associated reference are depicted in red to serve as an aid during construction.

The tool generates the graph following RCM construction principles, and therefore it will be a directed graph with only one main concept on the left side of the graph. The rest of the concepts are automatically located on the right of the main concept following the statement structure. When, in the opinion of the creator, the RCM is structurally complete, the user can assign layers to the included concepts.

5.2 References and Paths Interaction

One of the main features in RCMs is references management. Every definition and every path must have an associated reference. The graph reflects the fact that different paths can share concepts or linking phrases, so directed links between elements are labeled with the applicable references (see example in Fig. 5).

The tool manages a repository of references that the user can add in three different ways. Firstly, the user can add a reference manually in BibTeX format. Secondly, the references may come from a BibTeX file. Lastly, references can be imported from external repositories such as Mendeley (which requires a user account). OAuth 2.0 authentication is used in order that the user can give secure access to their saved references in Mendeley. This is an open authorisation standard, which provides client applications like RCMTool secure delegated access to server resources on behalf of a resource owner.

Each of the references in the RCMTool internal repository is identified by a number, and includes basic information about the source: title, authors, date, URL, etc. Whenever a reference is assigned to a set of correlative elements, one path is created. This is the case when branches of the graph are created from referenced statements by using the engine (see previous subsection). Paths must have a unique identifier that links them to the reference (reference identifier). If two paths are connected to the same reference (e.g., when the same source includes two different definitions of the same concept), the tool disambiguates them with a different label, such as the reference identifier and a letter of the alphabet (i.e. 1, 1b, 1c, etc.).

5.3 Example

Fig. 5 shows the tool in use with an RCM collecting definitions of the ‘Language’ concept. Since this is a highly complex concept, for the sake of brevity we present an RCM that includes only ten definitions from nine different references.

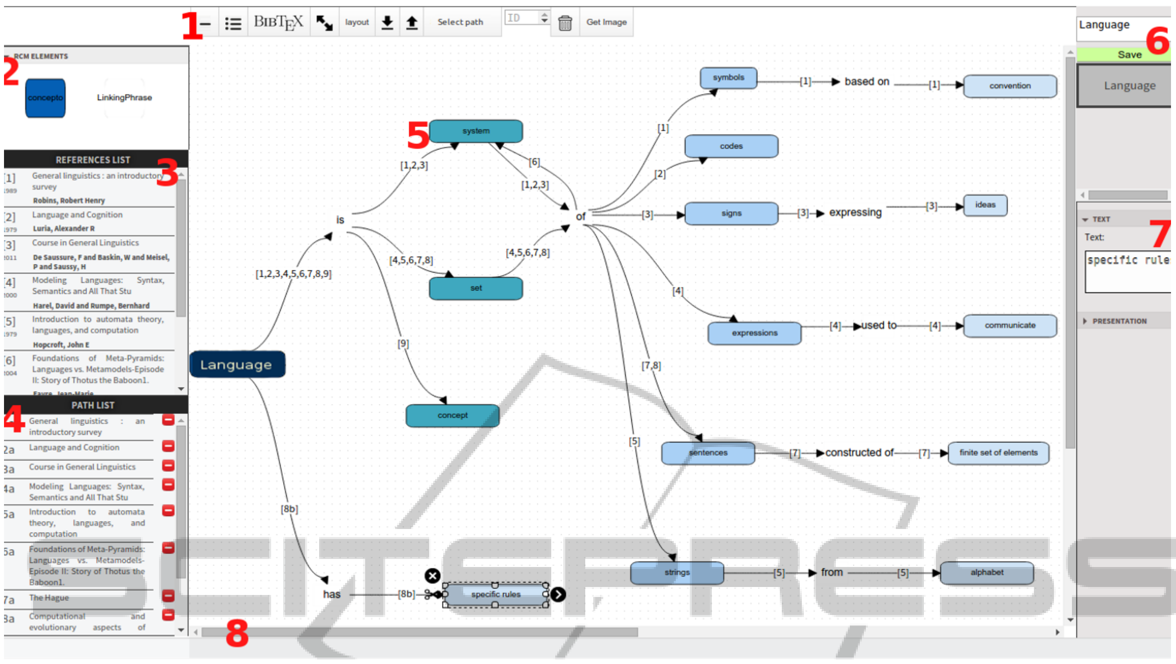
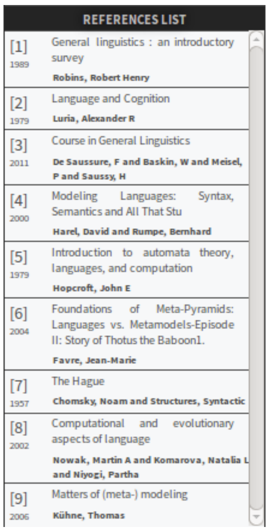


Figure 5: Tool capture with the RCM of ‘Language’. The areas marked with large red numbers correspond to the list in Section 4.2.


REFERENCES LIST

[1] 1989	General linguistics : an introductory survey **Robins, Robert Henry**
[2] 1979	Language and Cognition **Luria, Alexander R**
[3] 2011	Course in General Linguistics **De Saussure, F and Baskin, W and Meisel, P and Saussy, H**
[4] 2000	Modeling Languages: Syntax, Semantics and All That Stu **Harel, David and Rumpe, Bernhard**
[5] 1979	Introduction to automata theory, languages, and computation **Hopcroft, John E**
[6] 2004	Foundations of Meta-Pyramids: Languages vs. Metamodels-Episode II: Story of Thotus the Baboon1. **Favre, Jean-Marie**
[7] 1957	The Hague **Chomsky, Noam and Structures, Syntactic**
[8] 2002	Computational and evolutionary aspects of language **Nowak, Martin A and Komarova, Natalia L and Niyogi, Partha**
[9] 2006	Matters of (meta-) modeling **Kühne, Thomas**

Figure 6: References list (zoomed).

Following the procedures described in the previous sections, we have included the set of references, in this case by connecting the tool to Mendeley. The references list appears in area 3 of the interface, enlarged in Fig. 6 for better readability. The definitions that we want to represent are extracted from these references (each number in the following numbered list corresponds to the reference identifier in Fig. 6).

1. Language is a system of symbols based on convention.
2. Language is a system of codes.
3. Language is a system of signs expressing ideas.
4. Language is a set of expressions used to communicate.
5. Language is a set of strings from alphabet.
6. Language is a set of system.
7. Language is a set of sentences constructed of finite set of elements.
8. (a)Language is a set of sentences. (b)Language has specific rules.
9. Language is a concept.

These definitions are included in a list of statements, including a reference identifier for each statement. The tool processes the list, and generates a graph that can be revised for corrections. In the last step, the user must classify concepts in the four available layers (depicted with a color gradation in the graph).

6 CONCLUSIONS AND FURTHER WORK

We have presented RCMTool, a web application dedicated to the creation of References-enriched Concept

Maps. This technique allows to the compact representation of multiple definitions of the same concept, and the comparison of these definitions by using different criteria and metrics. These features are especially useful when modeling tasks are performed, in which there are often ambiguous concepts whose definition must be agreed by all members of the development team. Additionally, RCMs themselves are models, since they are created as instances of the metamodel that presented in Section 3.1. In this sense the RCM technique acts as a modeling language that allows RCM modeling, and therefore RCMTool is a tool that enables the creation of such models.

RCMTool is a web application ready for further expansion. Among future improvements we will consider, for example, the automated computation of metrics (Rodriguez-Priego et al., 2013), and the inclusion of an advanced authoring system that facilitates the collaborative use of the tool.

ACKNOWLEDGEMENTS

This work was partially funded by Project MTM2014-54151-P.

REFERENCES

Ausubel, D. P. (1963). *The psychology of meaningful verbal learning*. Grune & Stratton.

Ausubel, D. P., Novak, J. D., Hanesian, H., et al. (1968). *Educational psychology: A cognitive view*. Holt, Rinehart and Winston New York.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Comput. Linguist.*, 21(4):543–565.

Buzan, T. (2006). *Use Your Head*. Mind Set Series. BBC Active.

Cañas, A. J., Carff, R., Hill, G., Carvalho, M., Arguedas, M., Eskridge, T. C., Lott, J., and Carvajal, R. (2005). Concept maps: Integrating knowledge and information visualization. In *Knowledge and information visualization*, pages 205–219. Springer.

Client IO (2014). Rappid toolkit. http://jointjs.com/about-rappid. [Stable release: Rappid 1.4 / September 16, 2014].

FreeMind Dev. Team (2014). FreeMind tool. http://freemind.sourceforge.net/wiki/index.php/Main_Page/. [Stable release: FreeMind 1.0.1 / April 12, 2014].

Kowata, J. H., Cury, D., and Boeres, M. C. S. (2010). Concept maps core elements candidates recognition from text. In *Proceedings of Fourth International Conference on Concept Mapping*, pages 120–127, Viña del Mar, Chile.

Novak, J. D. (1984). *Learning how to learn*. Cambridge University Press.

Novak, J. D. and Cañas, A. J. (2006). The theory underlying concept maps and how to construct them. Technical report, Institute for Human and Machine Cognition, Pensacola Fl.

Park, J. and Hunting, S. (2003). *XML Topic Maps: Creating and Using Topic Maps for the Web*. Addison-Wesley.

Rodriguez-Priego, E., Garcia-Izquierdo, F. J., and Rubio, A. L. (2013). References-enriched Concept Map: a tool for collecting and comparing disparate definitions appearing in multiple references. *Journal of Information Science*, 39(6):789–804.

Santorini, B. (1990). Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision). Technical report, University of Pennsylvania.

XMind Ltd. (2014). XMind tool. http://www.xmind.net/. [Stable release: XMind 6 (v3.5.1) / 20th, Nov 2014].